

第6章 中央处理器

前言

中央处理器（CPU）是整个计算机的核心，它包括**运算器**和**控制器**。本章着重讨论CPU的功能和组成，控制器的工作原理和实现方法，微程序控制原理，基本控制单元的设计以及先进的CPU系统设计技术。



目录

1

6.1中央处理器的功能和组成

2

6.2控制器的组成和实现方法

3

6.3时序系统与控制方式

4

6.4微程序控制器



目录

1

6.1中央处理器的功能和组成

2

6.2控制器的组成和实现方法

3

6.3时序系统与控制方式

4

6.4微程序控制器



- 计算机的工作过程就是程序的运行过程，也就是在控制器的控制下逐条执行程序各指令的过程。在程序运行过程中，计算机的各部件在控制器的控制下有条不紊地工作，在各部件之间流动的指令和数据形成了指令流和数据流。





- 从程序运行的角度来看，控制器的基本功能是对指令流和数据流在时间与空间上实施正确的控制。
- 对指令流的控制：指令流出的控制;指令分析与执行的控制;指令流向的控制





- 对数据流的控制主要应包括对数据的流入与流出的控制；对数据变换、加工等操作的控制。
- 对于冯·诺依曼结构的计算机而言，数据流是根据指令流的操作而形成的，也就是说数据流是由指令流来驱动的。





- CPU中的主要寄存器是用来暂时保存在运算和控制过程中的中间结果、最终结果以及控制、状态信息的，它又可分为通用寄存器和专用寄存器两种。





- 1.通用寄存器
 - 通用寄存器可用来存放原始数据和运算结果，有的还可以作为变址寄存器、计数器、地址指针等。通用寄存器一般可以由CPU直接访问。





- 2.专用寄存器

- 专用寄存器是专门用来完成某一种特殊功能的寄存器。CPU中至少要有五个专用的寄存器。它们是：程序计数器（PC）、指令寄存器（IR）、存储器地址寄存器（MAR）、存储器数据寄存器（MDR）、状态标志寄存器（PSWR）。





- (1) 程序计数器 (PC)
 - 程序计数器用来存放正在执行的指令地址或接着要执行的下条指令地址。
- (2) 指令寄存器 (IR)
 - 指令寄存器用来存放从存储器中取出的指令。



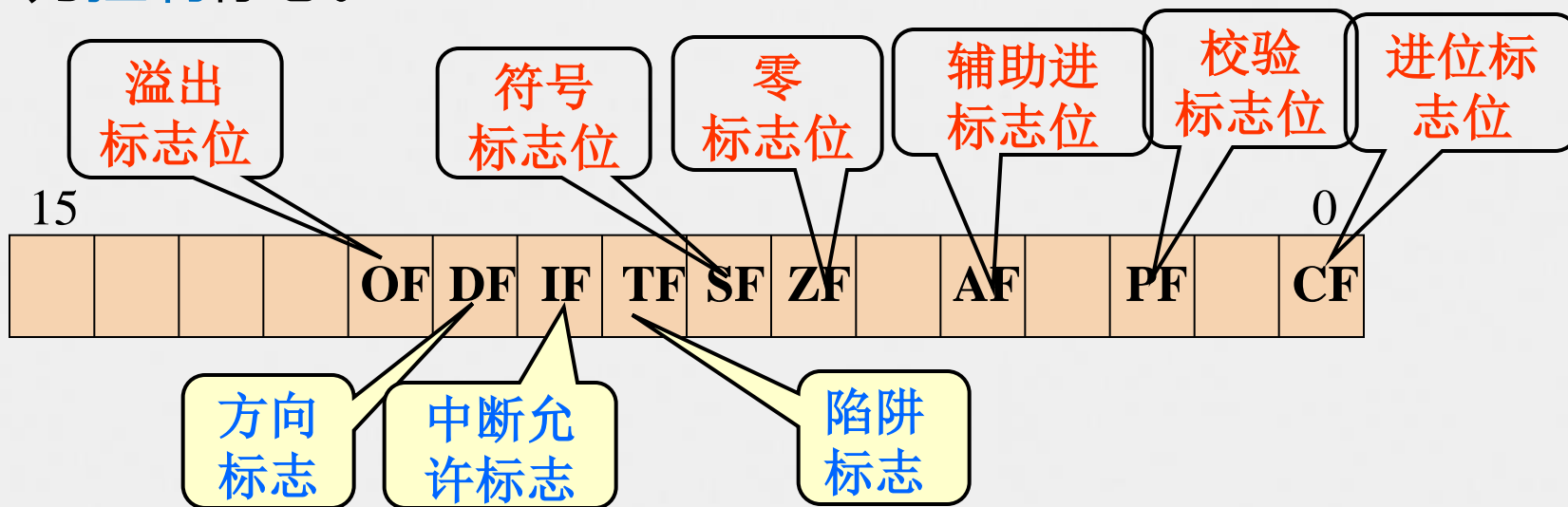


- (3) 存储器地址寄存器 (MAR)
 - 存储器地址寄存器用来保存当前CPU所访问的主存单元的地址。
- (4) 存储器数据寄存器 (MDR)
 - 存储器数据寄存器用来暂时存放由主存储器读出的一条指令或一个数据字；反之，当向主存存入一条指令或一个数据字时，也暂时将它们存放在存储器数据寄存器中。



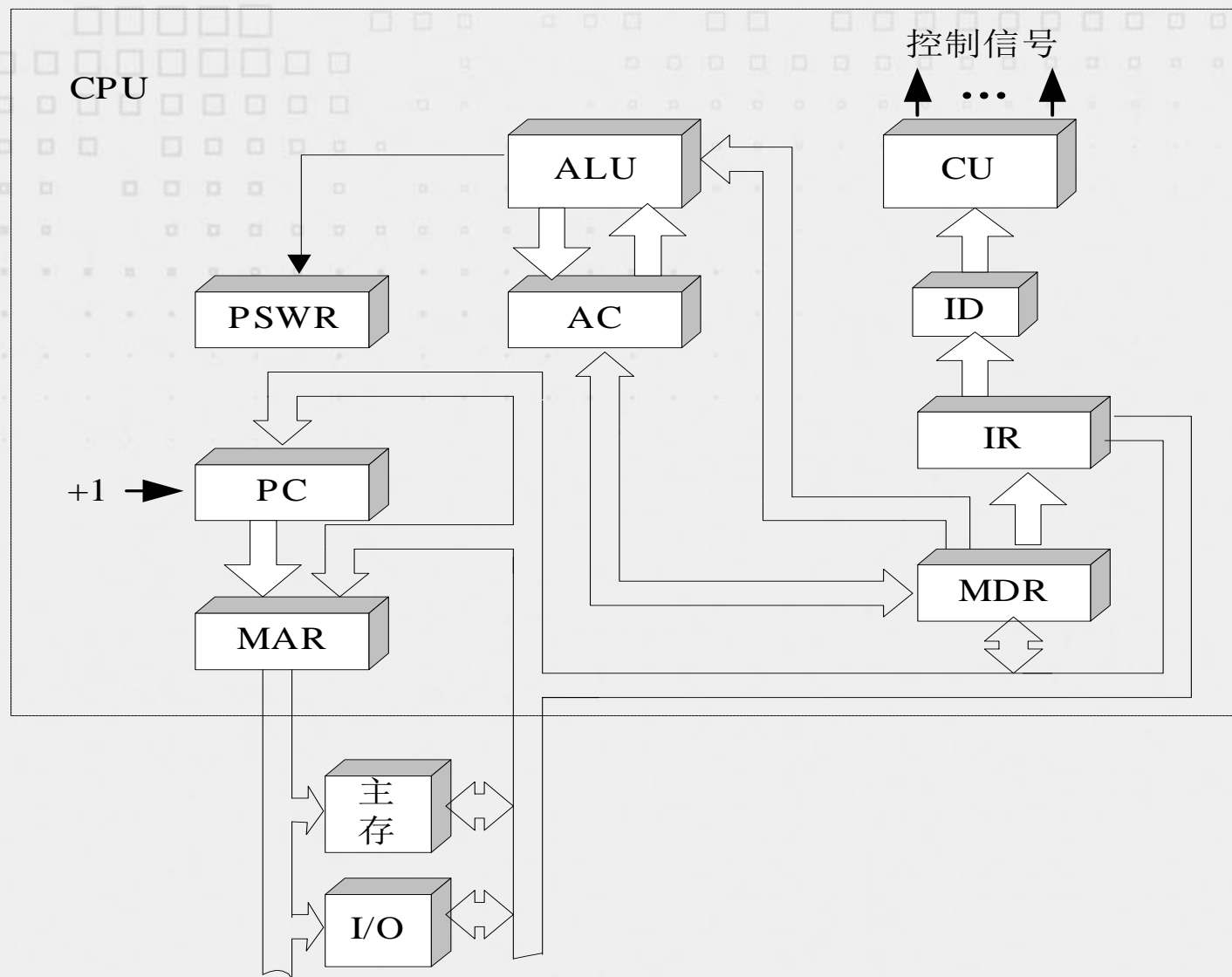


- (5) 状态标志寄存器 (PSWR)
 - 状态标志寄存器用来存放程序状态字的。程序状态字的各位表征程序和机器运行的状态，是参与控制程序执行的重要依据之一。
 - 8086的状态标志寄存器共16位，包括九个标志位，其中六个为状态标志，三个为控制标志。



6.1.3

CPU的组成

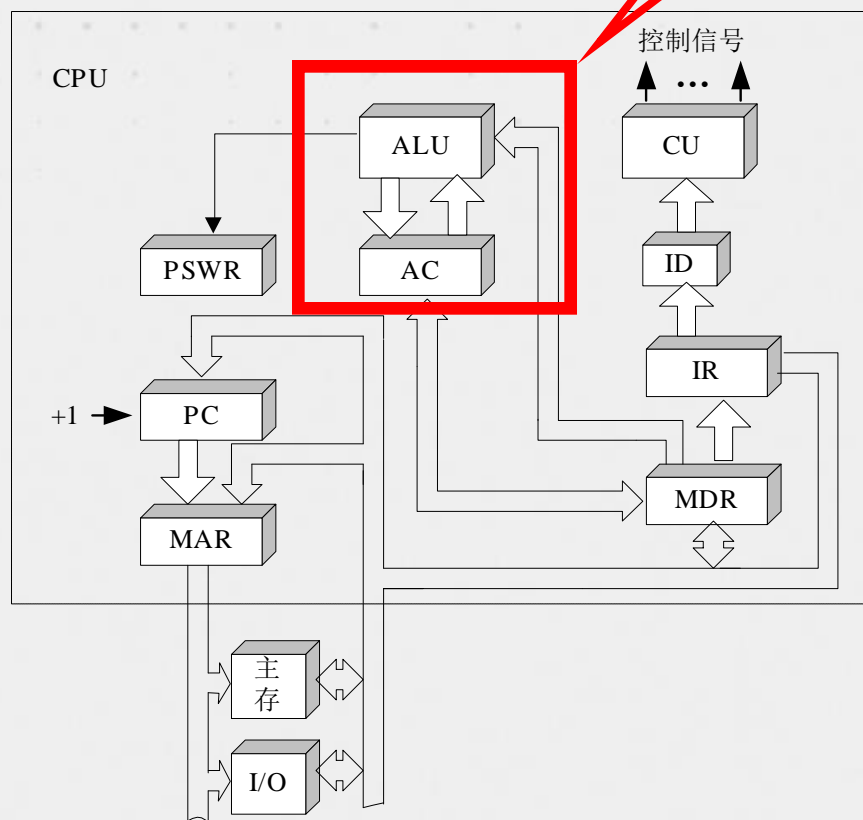


6.1.3

CPU的组成

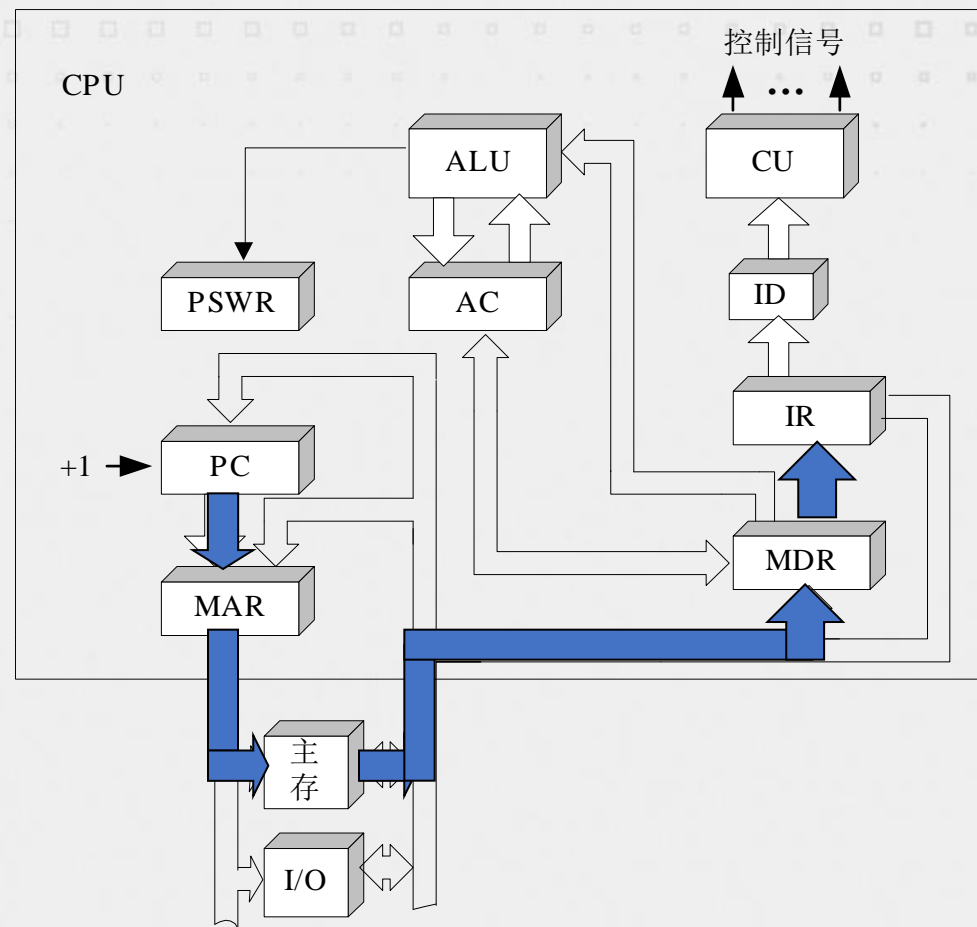
- 运算器的主要功能有：
 - (1)执行所有的算术运算；
 - (2)执行所有的逻辑运算，并进行逻辑测试。

运算器





- 控制器的主要功能有：
 - (1)从主存中取出一条指令，并指出下一条指令在主存中的位置。
 - (2)对指令进行译码或测试，产生相应操作控制信号，以便启动规定动作。
 - (3)指挥并控制CPU、主存和输入/输出设备之间的数据流动方向。





- 例：CPU结构如图所示，其中有一个累加器AC、一个状态条件寄存器和其他四个寄存器，各部分之间的连线表示数据通路，箭头表示信息传送方向。
 - （1）表明图中四个寄存器的名称。
 - （2）简述指令从主存取到控制器的数据通路。
 - （3）简述数据在运算器和主存之间进行存取访问的数据通路。



练习



CPU

主存储器M

a

c

AC

b

ALU

d

+1

状态寄存器

操作
控制器

• 指令通路：

– PC → MAR → 主存M → MDR → 指令寄存器
IR → 操作控制器

• 数据通路：

– 读数据

MAR先置数据地址，M → MDR → ALU → AC

– 写数据

MAR先置数据地址，AC → MDR → M





- 1.字长
- 2.内部工作频率
- 3.外部工作频率
- 4.前端总线频率
- 5.QPI数据传输速率
- 6.....





目录

1

6.1中央处理器的功能和组成

2

6.2控制器的组成和实现方法

3

6.3时序系统与控制方式

4

6.4微程序控制器

前言

控制器是计算机系统的指挥中心，它把运算器、存储器、输入/输出设备等部件组成一个有机的整体，然后根据指令的要求指挥全机的工作。

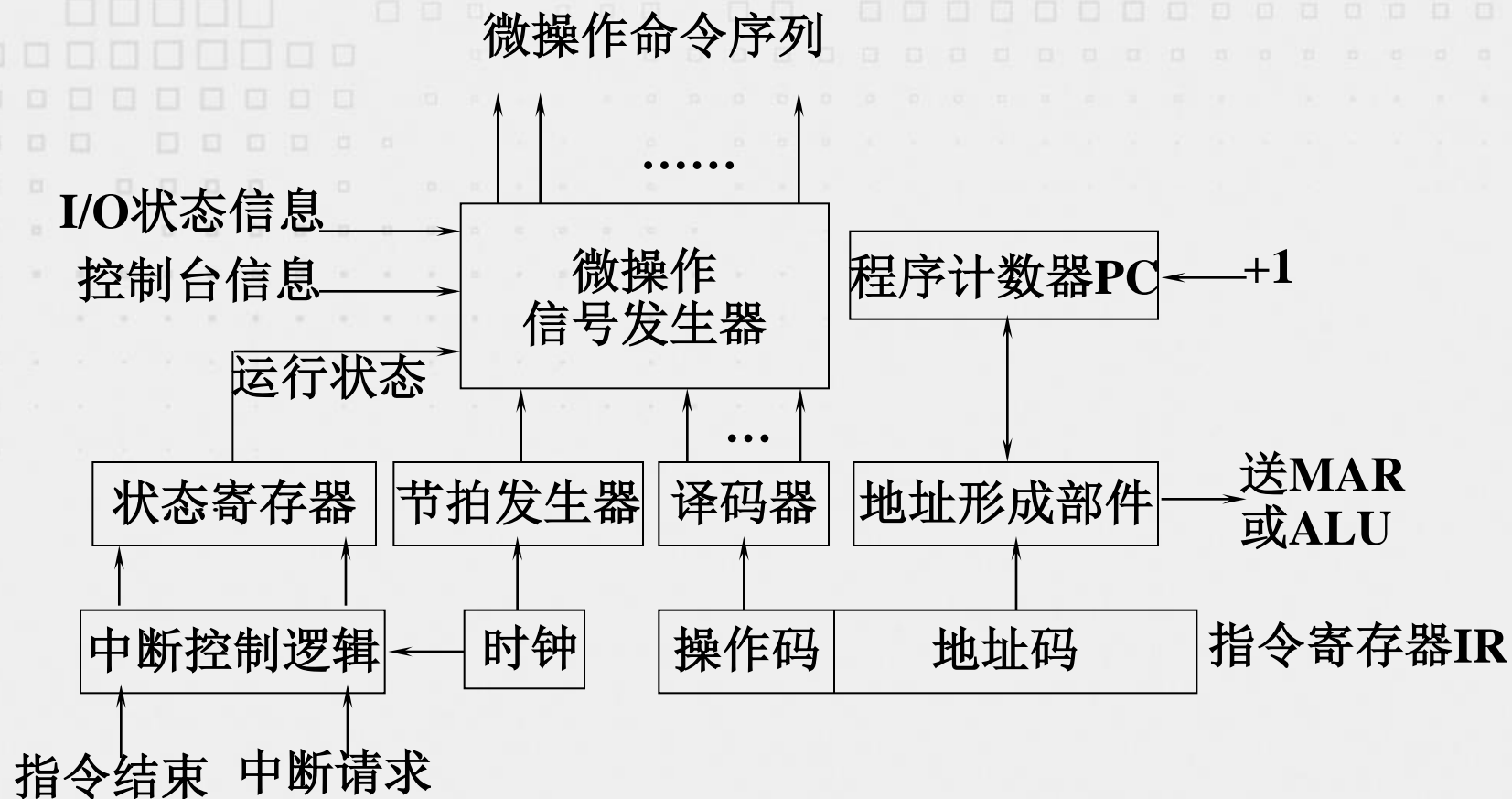


- 1.指令部件
- 2.时序部件
- 3.微操作信号发生器
- 4.中断控制逻辑



6.2.1

控制器的基本组成



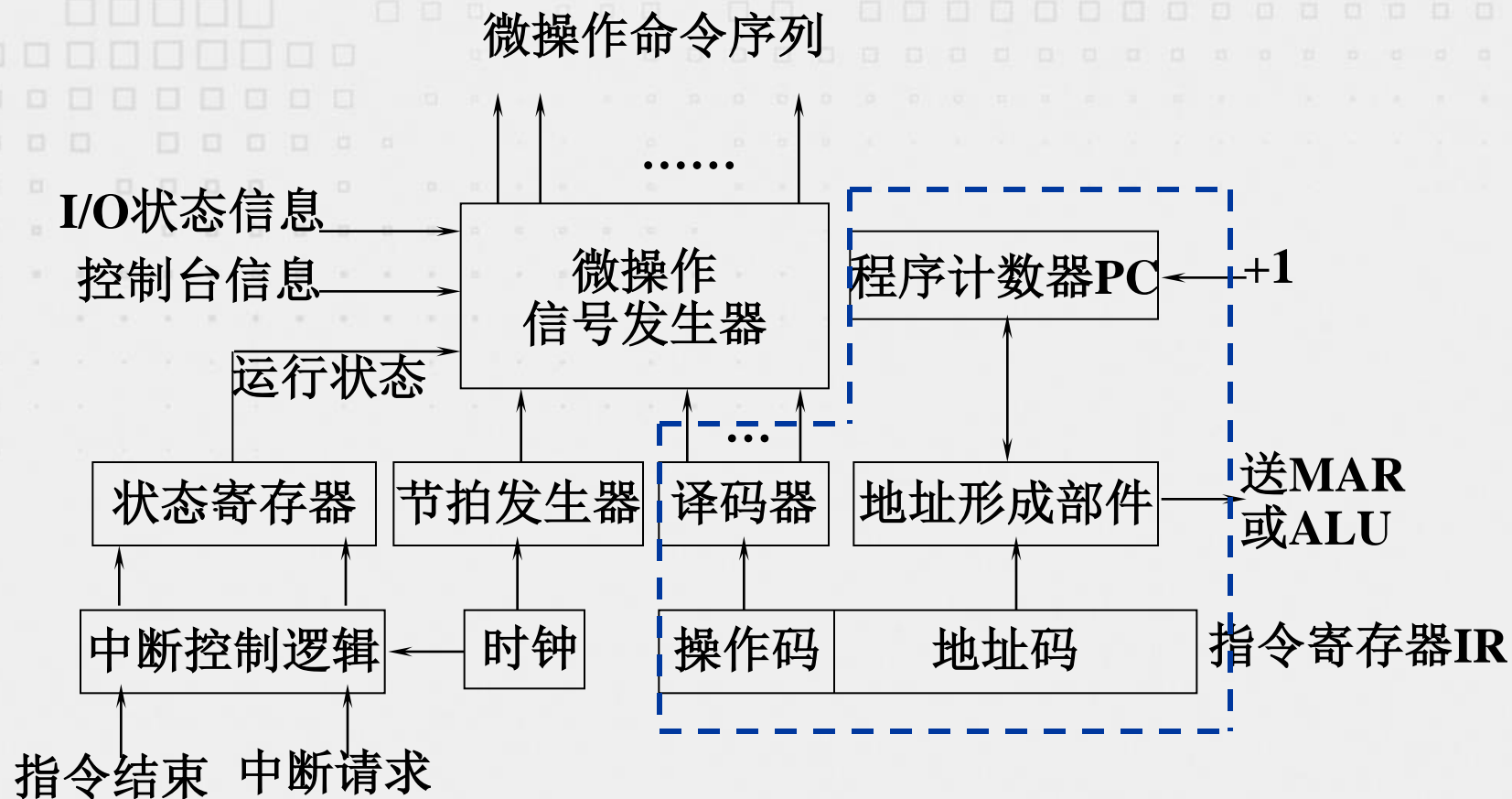


- 1.指令部件
 - 指令部件的主要任务是完成取指令并分析指令。指令部件包括：
 - ✎ (1) 程序计数器 (PC)
 - ✎ (2) 指令寄存器 (IR)
 - ✎ (3) 指令译码器 (ID)
 - ✎ (4) 地址形成部件



6.2.1

控制器的基本组成





- 2.时序部件

- 时序部件能产生一定的时序信号，以保证机器的各功能部件有节奏地进行

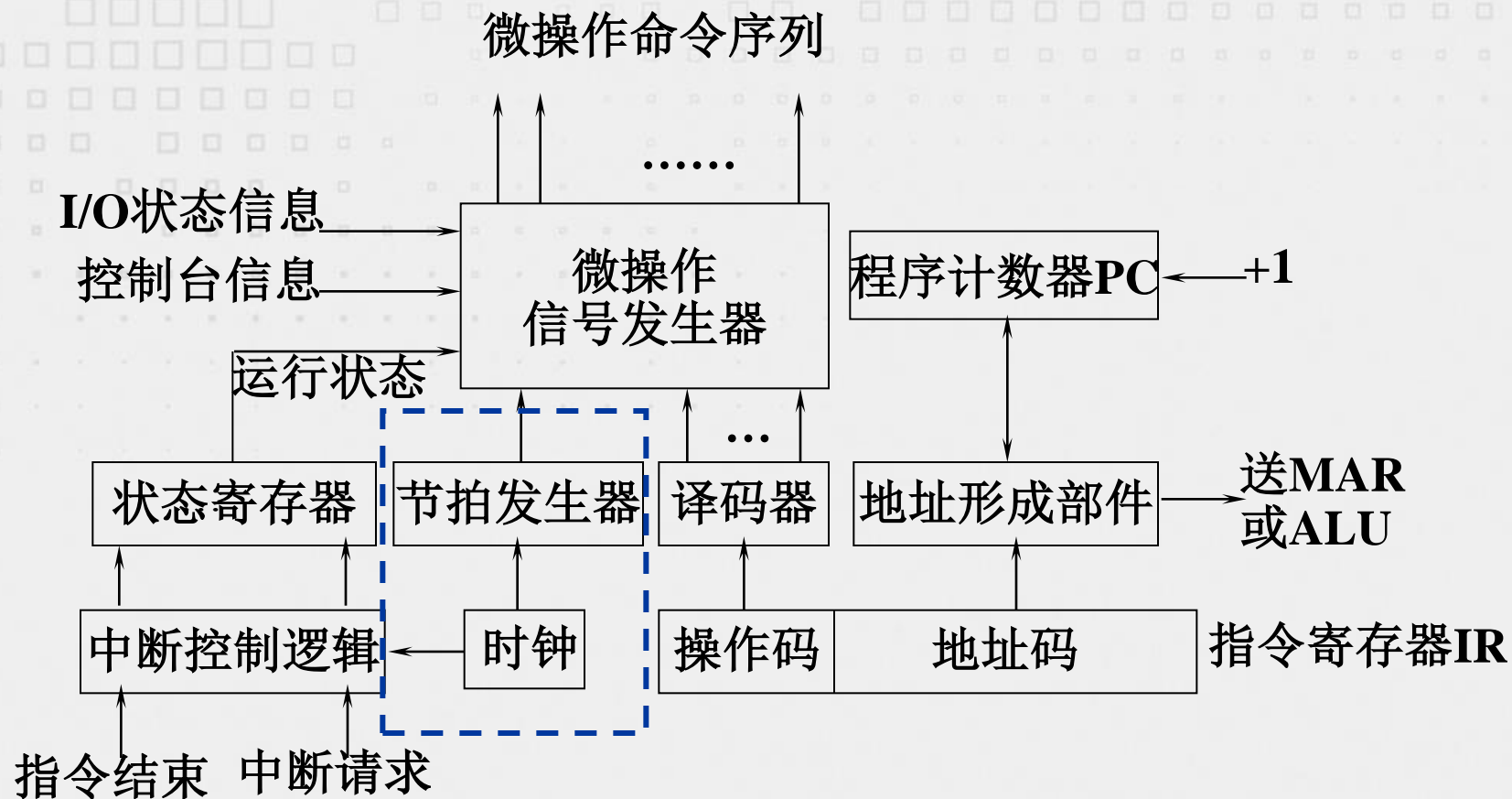
信息传送、加工及信息存储。时序部件包括：

- ✕ (1)脉冲源
 - ✕ (2)节拍信号发生器
 - ✕ (3)启停控制逻辑



6.2.1

控制器的基本组成





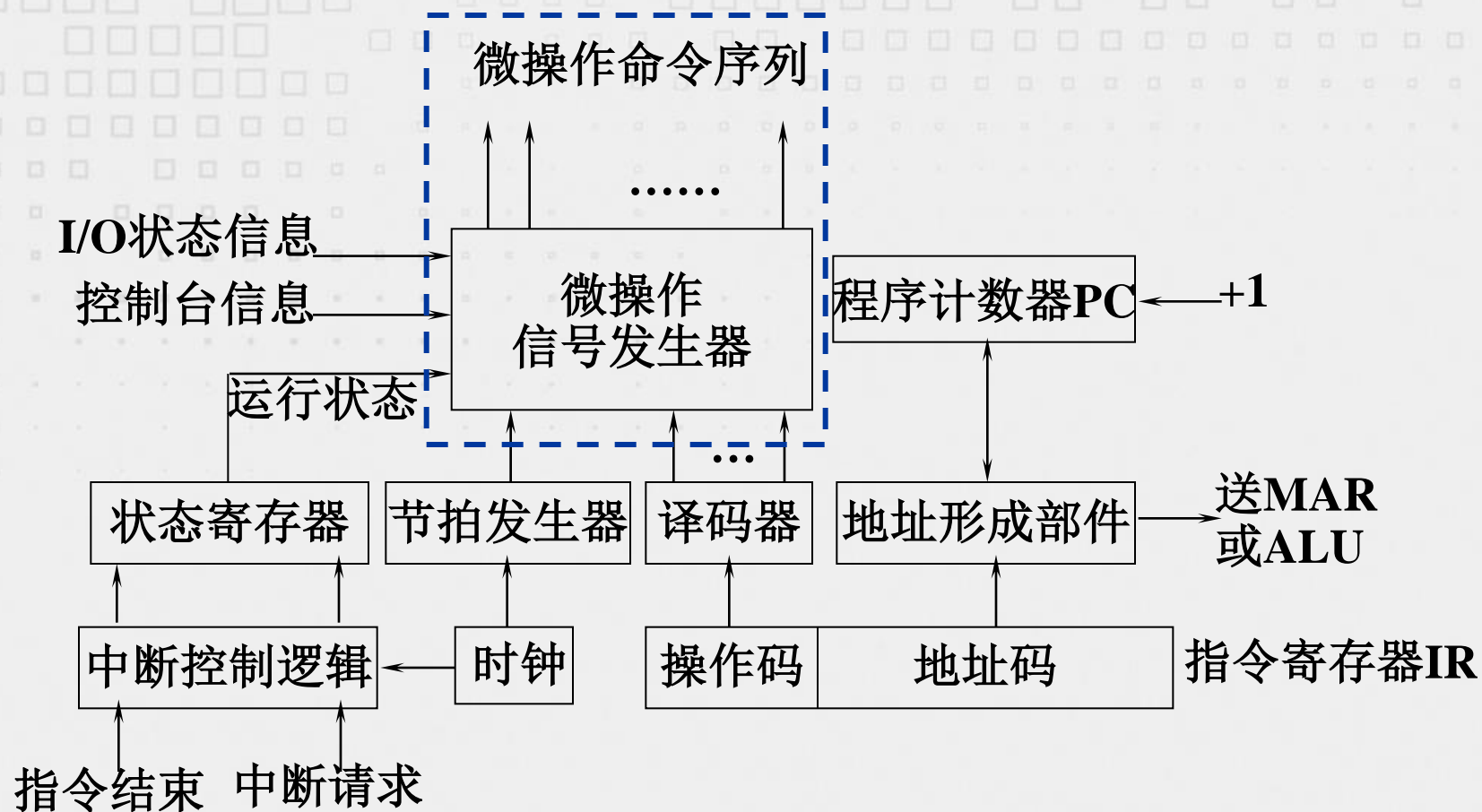
- 3.微操作信号发生器

- 一条指令的取出和执行可以分解成很多最基本的操作，这种最基本的不可再分割的操作称为微操作。微操作信号发生器也称为控制单元（CU）。
不同的机器指令具有不同的微操作序列。



6.2.1

控制器的基本组成



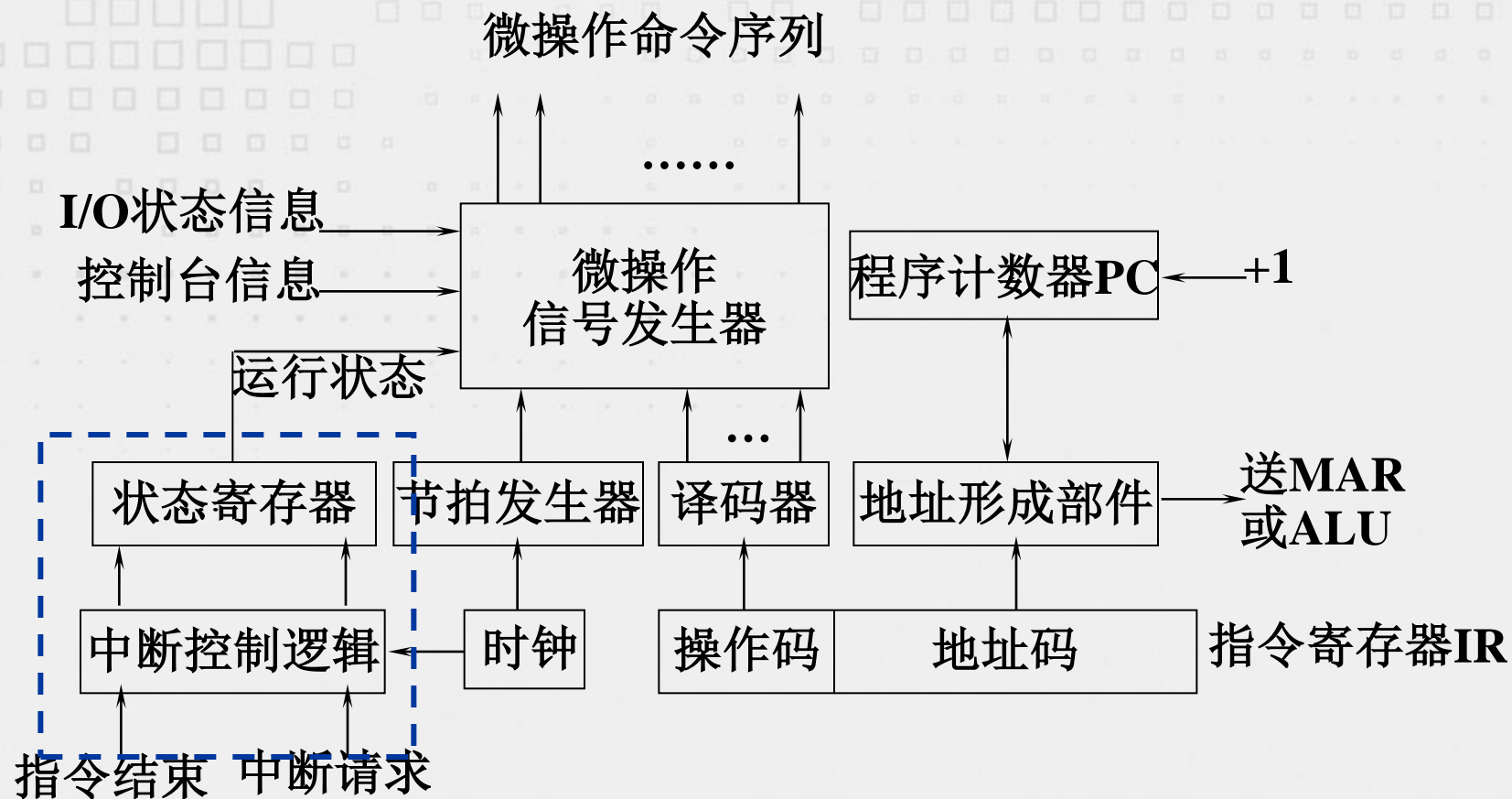


- 4.中断控制逻辑
 - 中断控制逻辑是用来控制中断处理的硬件逻辑。



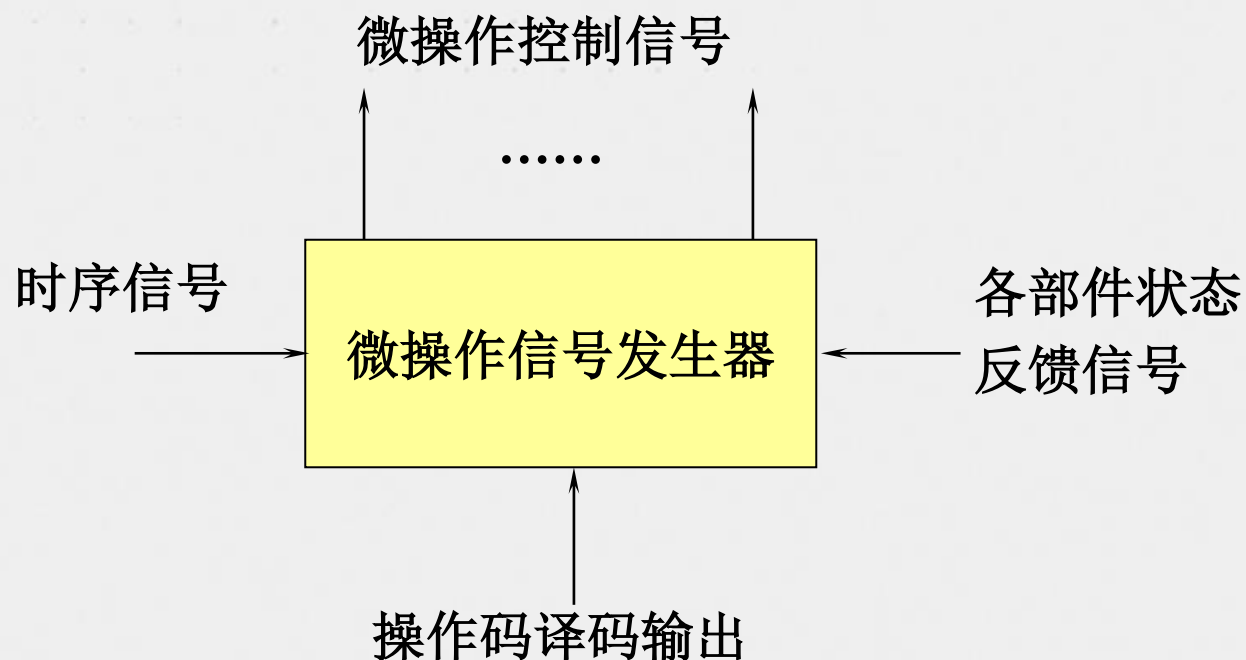
6.2.1

控制器的基本组成





- 控制器的输入是机器指令代码，输出是微操作控制信号，因此微操作信号发生器是控制器的核心。





- 根据产生微操作控制信号的方式不同，控制器可分为3种，它们的根本区别在于微操作信号发生器实现方法不同，而控制器中的其它部分基本上是大同小异的。
- 1.组合逻辑型
 - 这种控制器称为常规控制器或硬布线控制器，它是采用组合逻辑技术来实现的，其微操作序列形成部件是由门电路组成的复杂树形网络





- 2.存储逻辑型
 - 这种控制器称为微程序控制器，它是采用存储逻辑来实现的，也就是把微操作信号代码化，使每条机器指令转化成为一段微程序并存入一个专门的存储器（控制存储器）中，微操作控制信号由微指令产生。





- 3.组合逻辑和存储逻辑结合型
 - 这种控制器称为PLA控制器，它是组合逻辑技术和存储逻辑技术结合的产物，它克服了两者的缺点，是一种较有前途的方法。





目录

1

6.1中央处理器的功能和组成

2

6.2控制器的组成和实现方法

3

6.3时序系统与控制方式

4

6.4微程序控制器

前言

时序系统是控制器的核心，其功能是为指令的执行提供各种定时信号。



- 1.指令周期和机器周期
 - 指令周期是指取指令、分析指令到执行完该指令所需的全部时间。
 - 机器周期通常又称CPU周期，通常把一条指令划分为若干个机器周期，每个机器周期完成一个基本操作。





- 指令周期 = $i \times$ 机器周期

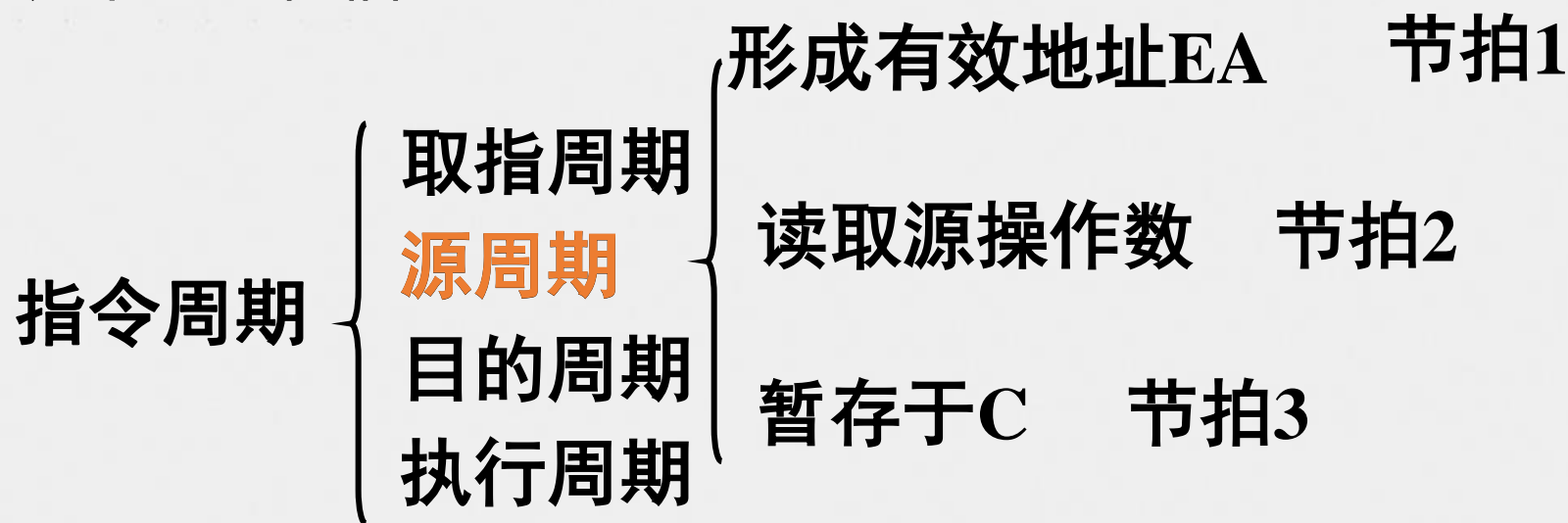
指令周期 { 取指周期
源周期
目的周期
执行周期





- 2.节拍

- 把一个机器周期分为若干个相等的时间段，每一个时间段对应一个电位信号，称为节拍电位信号。





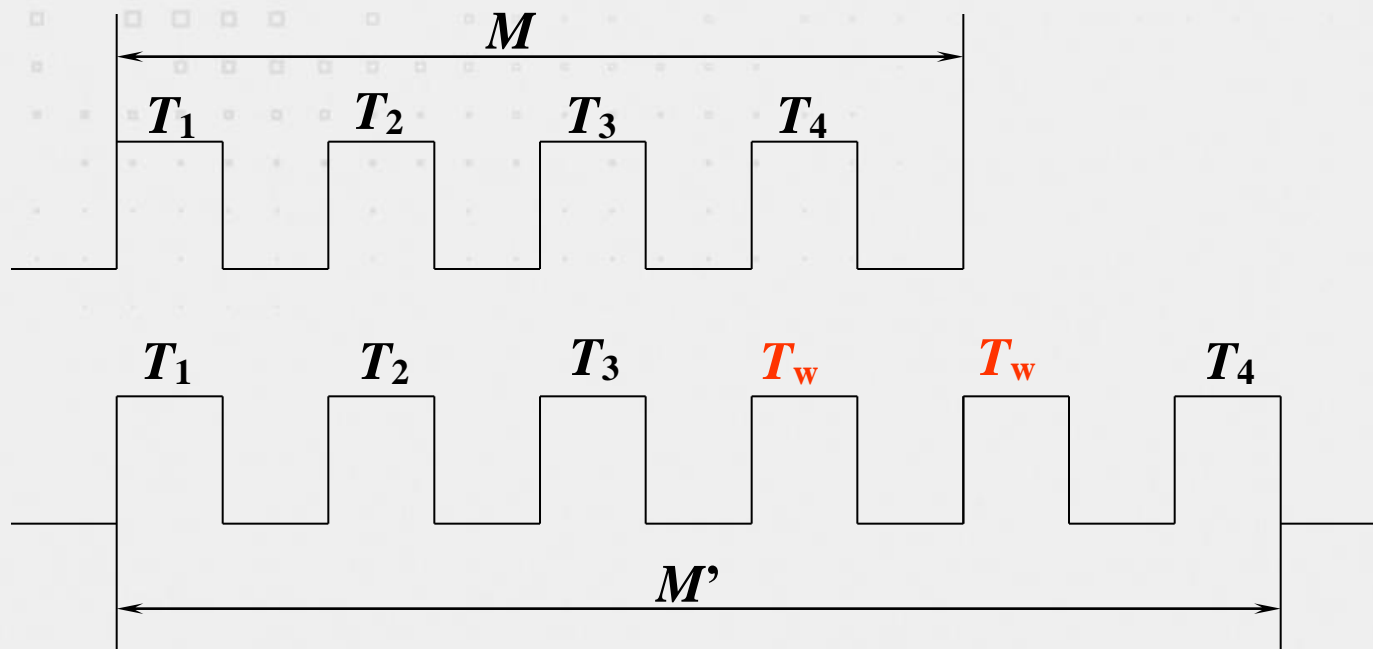
- 由于不同的机器周期内需要完成的微操作内容和个数是不同的，因此，不同机器周期内所需要的节拍数也不相同。节拍的选取一般有以下几种方法：
 - (1)统一节拍法-定长CPU周期。
 - (2)分散节拍法-不定长CPU周期
 - (3)延长节拍法
 - (4)时钟周期插入





6.3.1

时序系统





- 3.工作脉冲
 - 在节拍中执行的有些操作需要同步定时脉冲，为此，在一个节拍内常常设置一个或几个工作脉冲，作为各种同步脉冲的来源。工作脉冲的宽度只占节拍电位宽度的 $1/n$ ，并处于节拍的末尾。

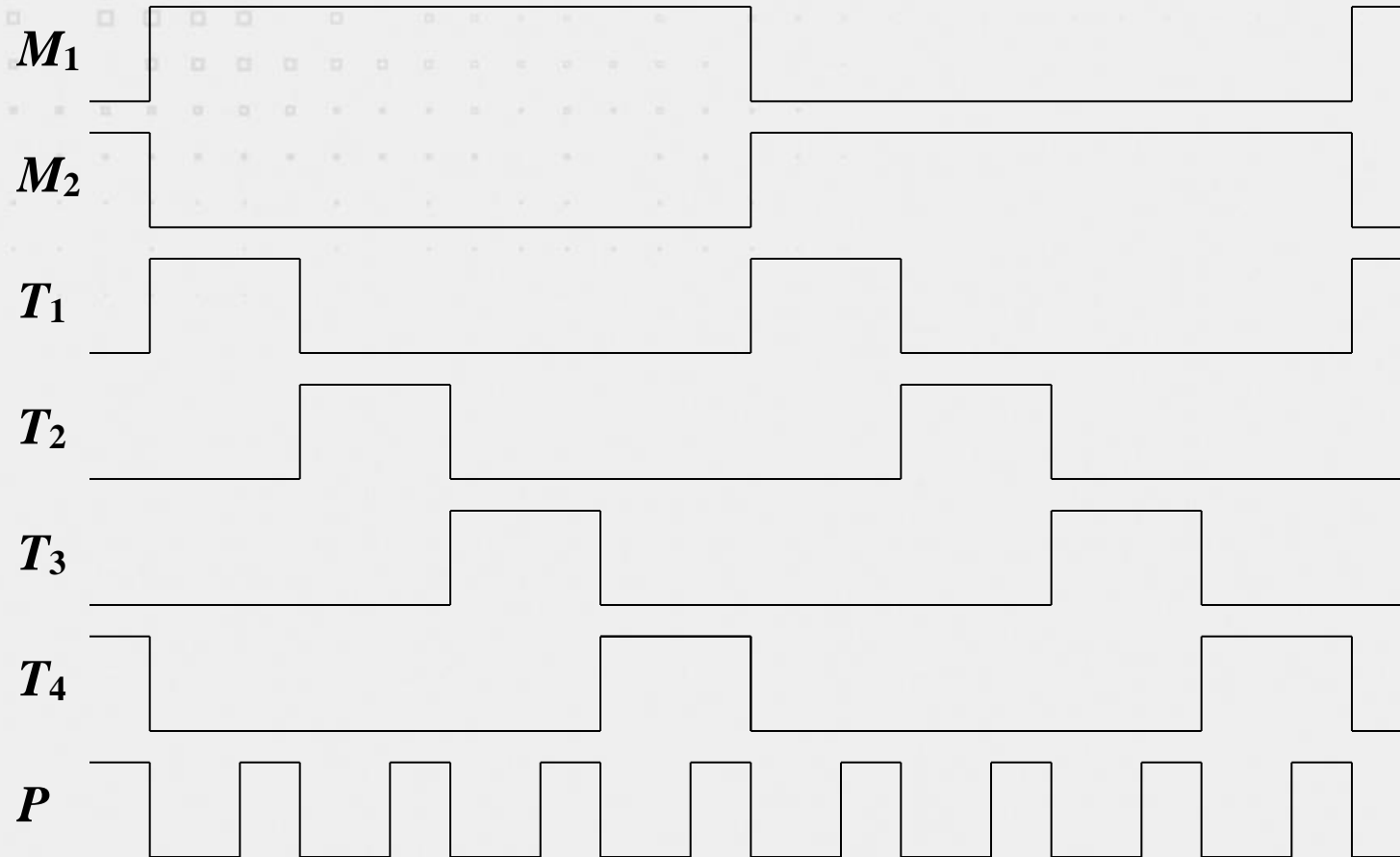




6.3.1

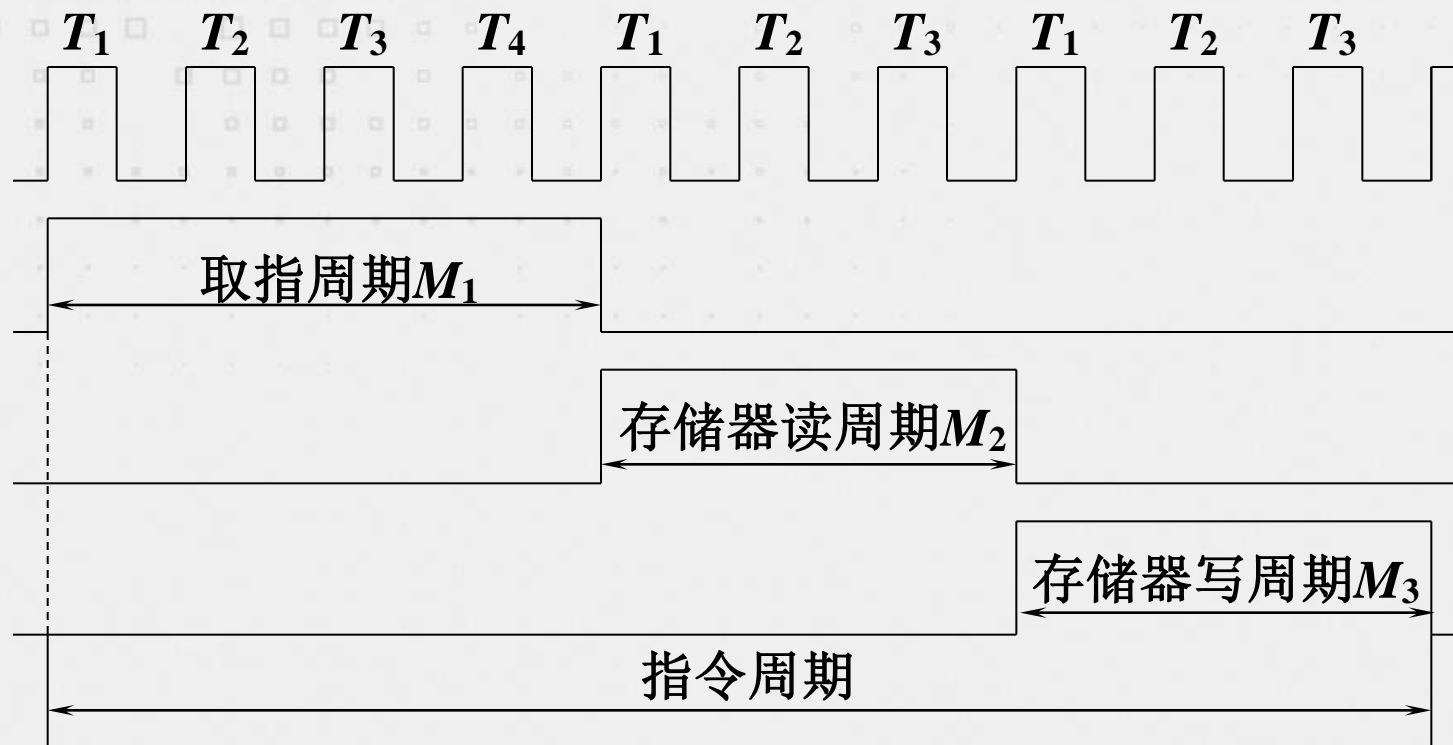
时序系统

- 4.多级时序系统





6.3.1 时序系统





- 1.同步控制方式
 - 同步控制方式即固定时序控制方式，各项操作都由统一的时序信号控制，在每个机器周期中产生统一数目的节拍电位和工作脉冲。由于不同的指令，操作时间长短不一致，同步控制方式应以最复杂指令的操作时间作为统一的时间间隔标准。





- 2.异步控制方式
 - 异步控制方式即可变时序控制方式。各项操作不采用统一的时序信号控制，而根据指令或部件的具体情况决定，需要多少时间，就占用多少时间。





- 3.联合控制方式
 - 这是同步控制和异步控制相结合的方式。实际上现代计算机中几乎没有完全采用同步或完全采用异步的控制方式，大多数是采用联合控制方式。通常的设计思想是：在功能部件内部采用同步方式或以同步方式为主的控制方式，在功能部件之间采用异步方式。



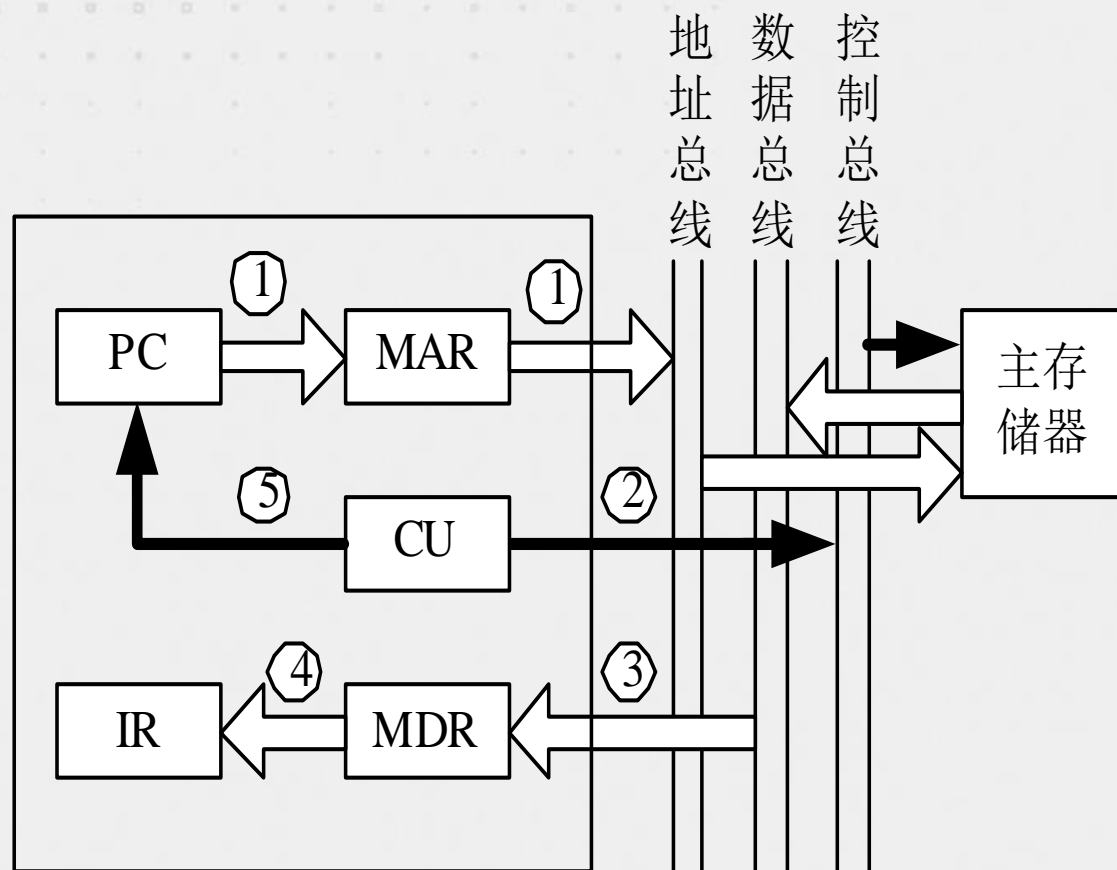


- 一条指令执行过程可以分为三个阶段：**取指令阶段**、**分析取数阶段**和**执行阶段**。
- 1.取指令阶段
 - 取指令阶段完成的任务是将现行指令从主存中取出来并送至指令寄存器中去。





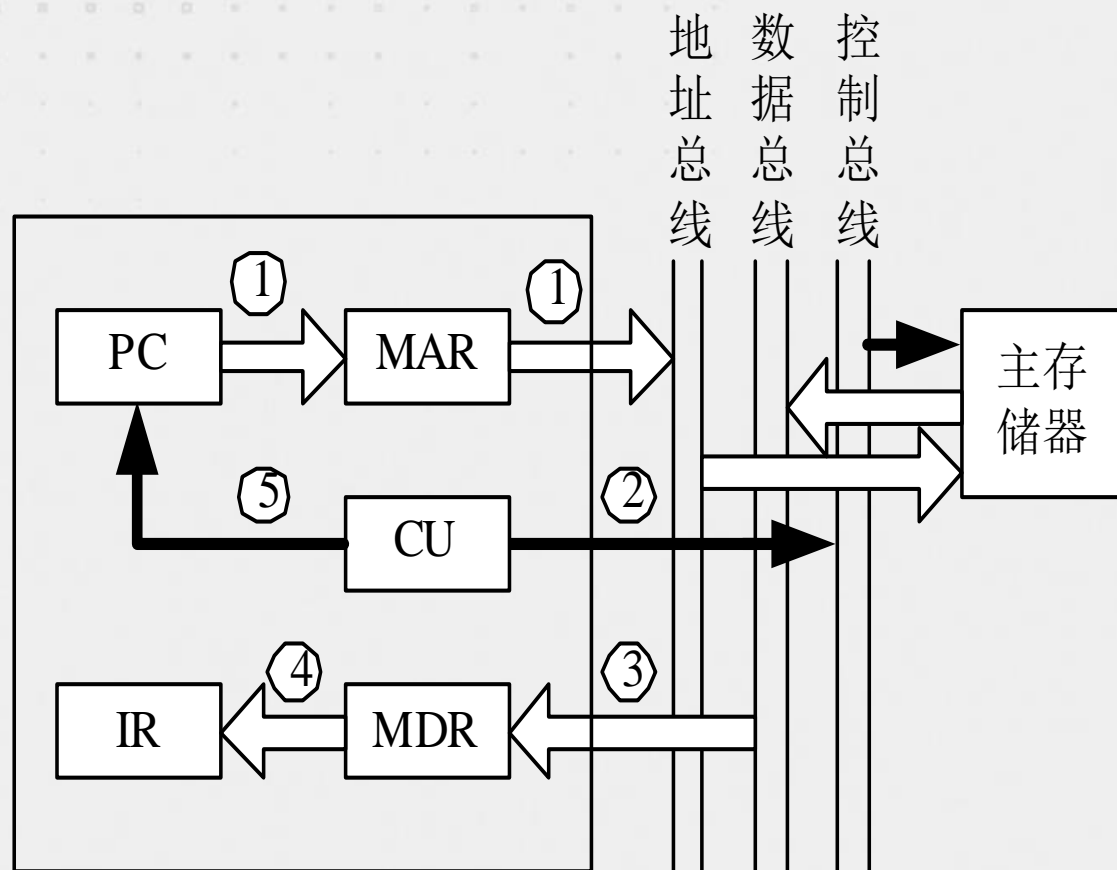
- ①将程序计数器（PC）中的内容送至存储器地址寄存器（MAR），并送地址总线（AB）。(PC)→MAR





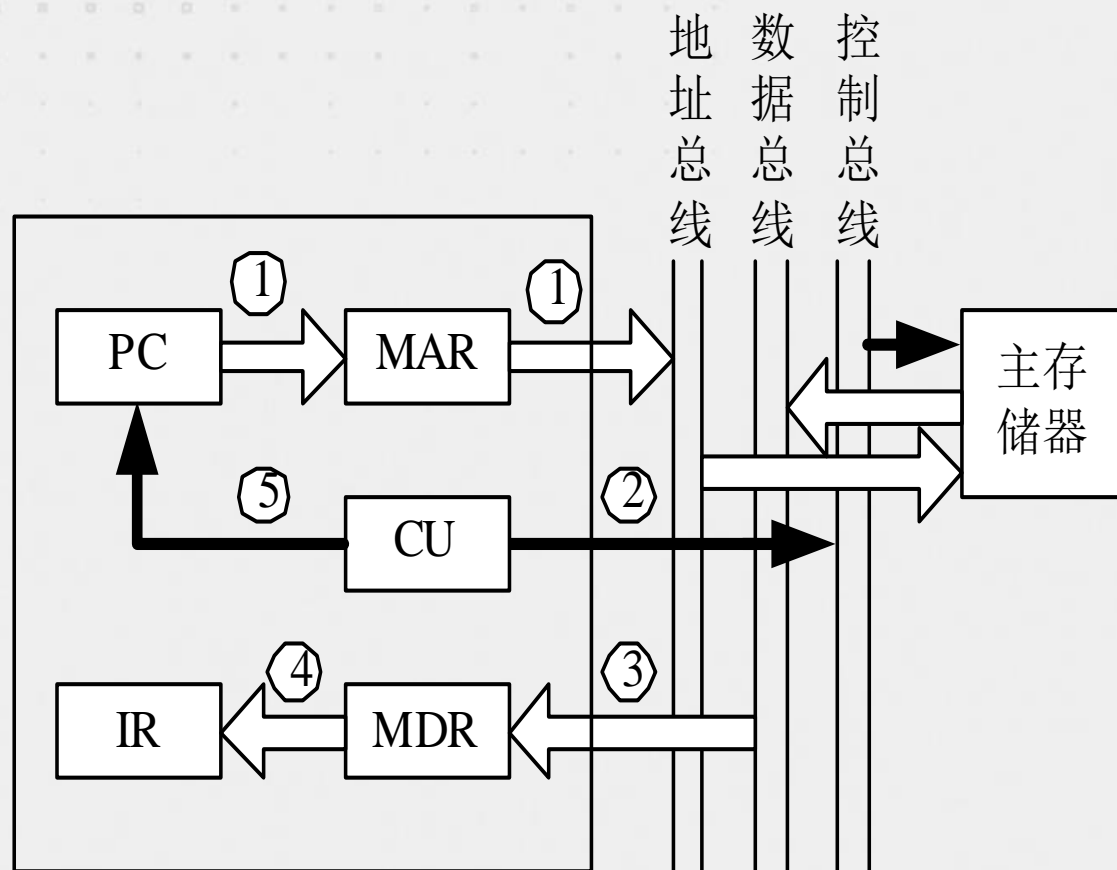
- ②由控制单元（CU）经控制总线（CB）向主存发读命令。

Read



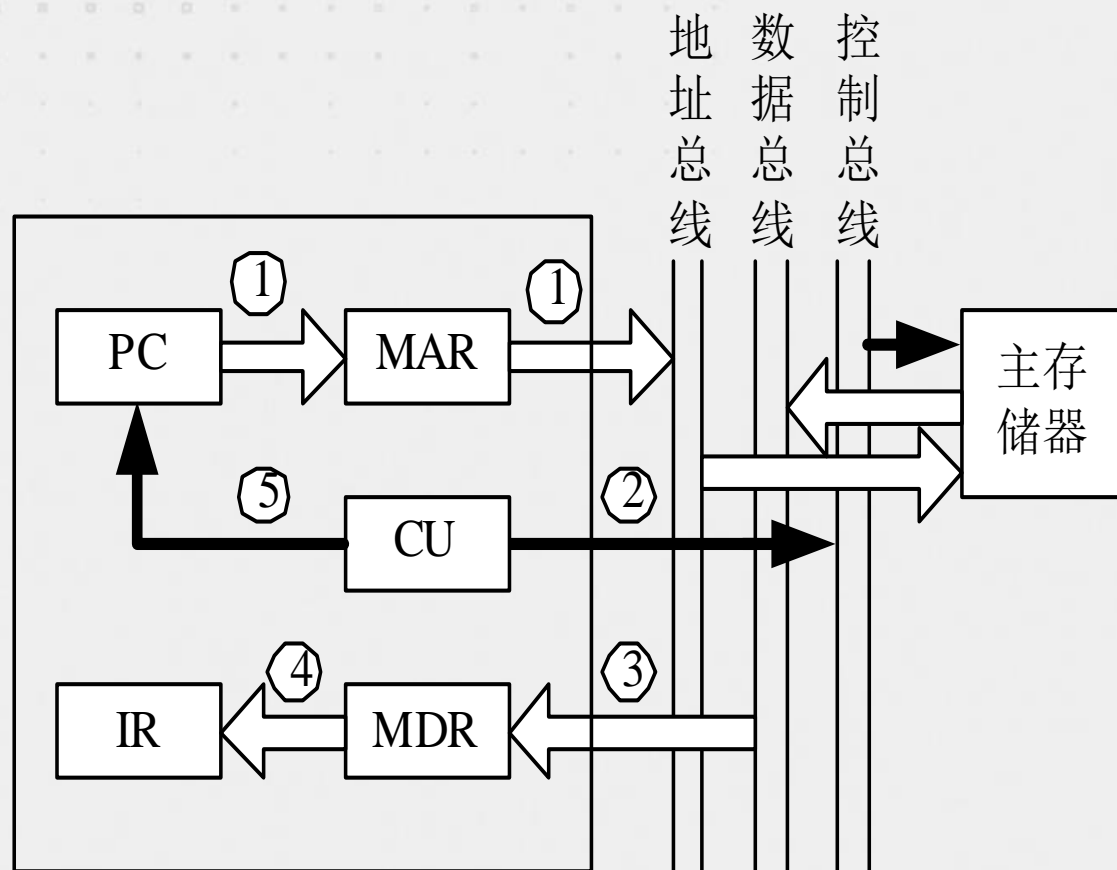


- ③从主存中取出的指令通过数据总线（DB）送到存储器数据寄存器（MDR）。 $M(MAR) \rightarrow MDR$



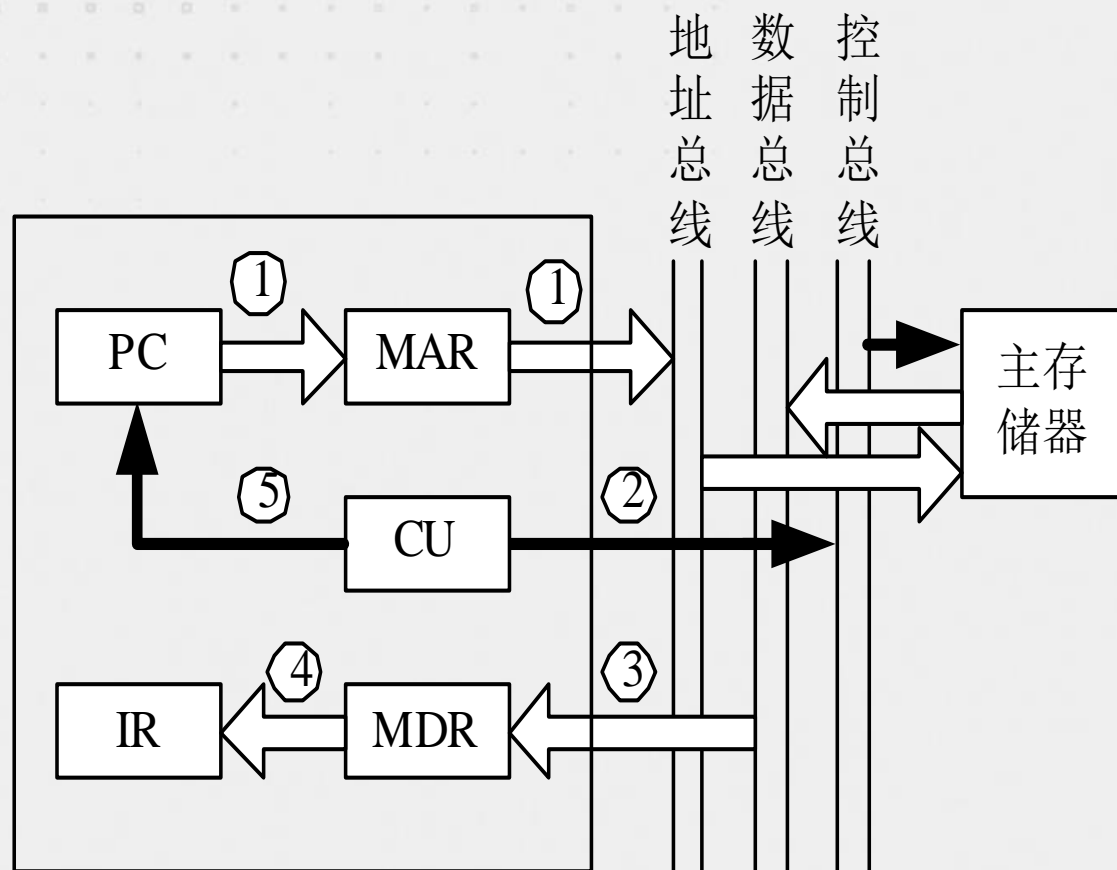


- ④将MDR的内容送至指令寄存器（IR）中。 (MDR)→IR



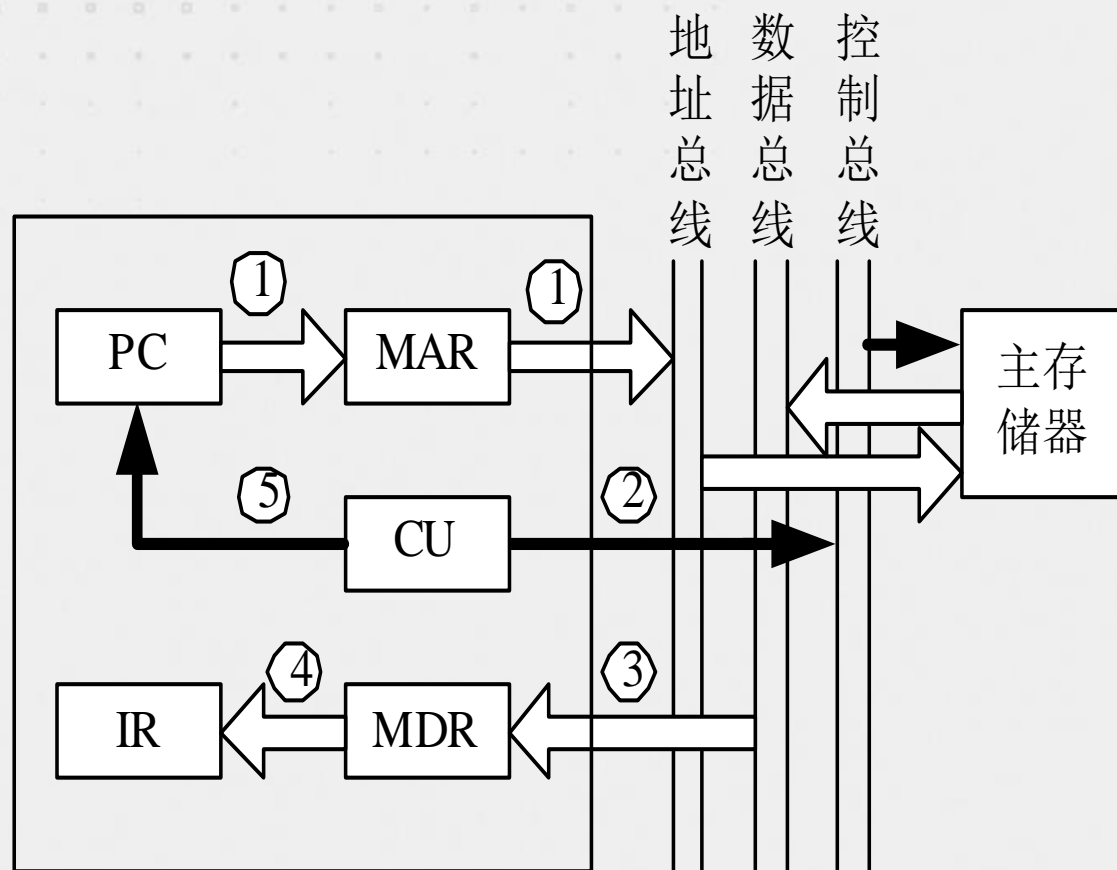


- ⑤将PC的内容递增，为取下一条指令做好准备。 $(PC) + 1 \rightarrow PC$





- 以上这些操作对任何一条指令来说都是必须要执行的操作，所以称为**公共操作**。





- 2.分析取数阶段
 - 取出指令后，机器立即进入分析指令阶段，指令译码器ID可识别和区分不同的指令类型及各种获取操作数的方法。由于各条指令功能不同，寻址方式也不同，所以分析取数阶段的操作是各不相同的。





- 3.执行阶段
 - 执行阶段完成指令规定的各种操作，形成稳定的运算结果，并将其存储起来。
 - 计算机的基本工作过程可以概括成为取指令、分析取数、执行指令，然后再取下一条指令，……。如此周而复始，直至遇到停机指令或外来的干预为止。





- 控制器在实现一条指令的功能时，总要把每条指令分解成为一系列时间上先后有序的最基本、最简单的微操作，即微操作序列。

微操作序列是与CPU的内部数据通路密切相关的，不同的数据通路就有不同的微操作序列。





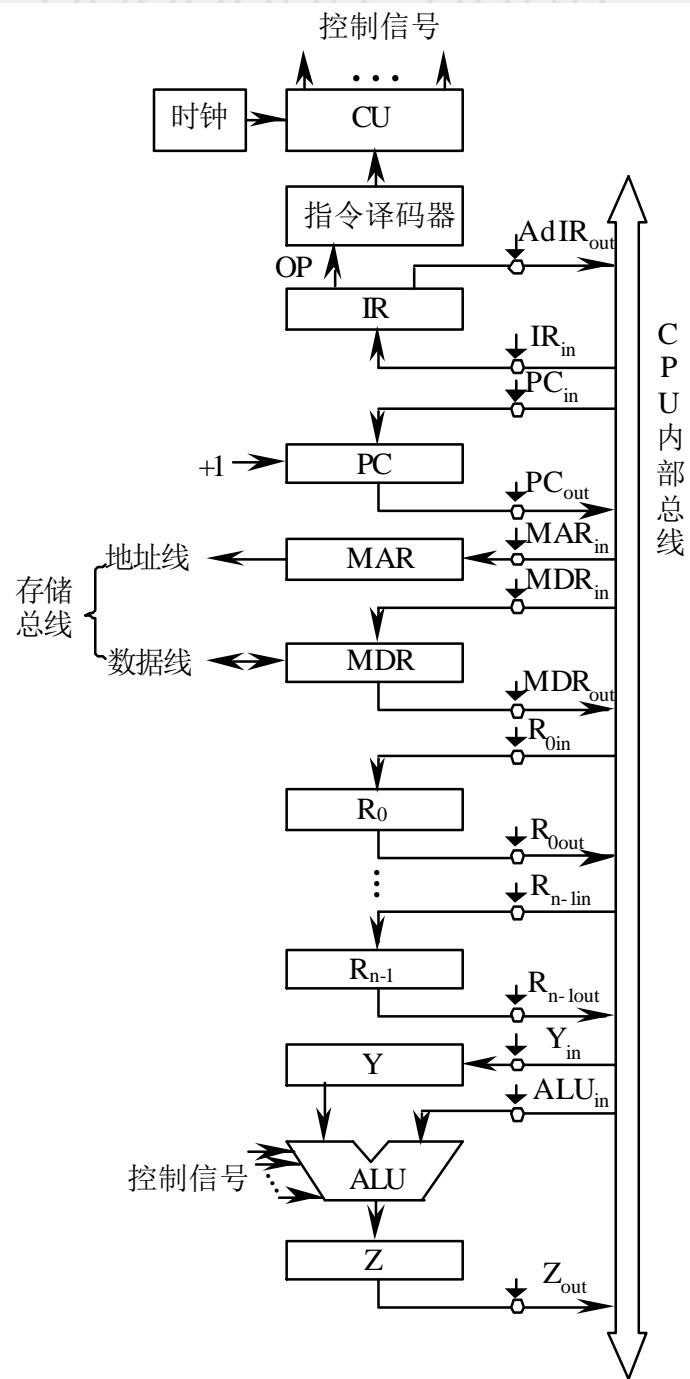
- 1.加法指令ADD R1,@R0
 - 这条指令完成的功能是把R0的内容作为地址送到主存以取得第一操作数，再与R1的内容相加，最后将结果送回主存中。即实现：
$$((R0)) + (R1) \rightarrow (R0)$$



6.3.4

指令的微操作序列

- 字母加下标in表示该部件的接收控制信号，实际上就是该部件的**输入开门**信号；字母加下标out表示该部件的发送控制信号，实际上就是该部件的**输出开门**信号。





- (1)取指周期
 - ① PC_{out} 和 MAR_{in} 有效，完成PC经CPU内部总线送至MAR的操作，记作 $(PC) \rightarrow MAR$ ；
 - ② 通过控制总线（图中未画出）向主存发读命令，记作Read；
 - ③ 存储器通过数据总线将MAR所指单元的内容（指令）送至MDR，记作 $M(MAR) \rightarrow MDR$ ；





- (1)取指周期
 - ④ MDR_{out} 和 IR_{in} 有效，将MDR的内容送至IR，记作 $(MDR) \rightarrow IR$ 。至此，指令被从主存中取出，其操作码字段开始控制CU。
 - ⑤ 使PC内容加1，记作 $(PC)+1 \rightarrow PC$ 。





- 这条指令的微操作序列的第① ~ ⑤步为取指令阶段的公共操作，

它完成的任务为：

- $(PC) \rightarrow MAR$
- Read
- $M(MAR) \rightarrow MDR \rightarrow IR$
- $(PC) + 1 \rightarrow PC$





- (2)取数周期
 - 取数周期要完成取操作数的任务，被加数在主存中，加数已放在通用寄存器 R_1 中。
 - ① R_{0out} 和 $MARin$ 有效，完成将被加数地址送至MAR的操作，记作 $(R_0) \rightarrow MAR$ ；
 - ② 向主存发读命令，记作 $Read$ ；
 - ③ 存储器通过数据总线将MAR所指单元的内容（数据）送至MDR，同时 MDR_{out} 和 Y_{in} 有效，记作 $M(MAR) \rightarrow MDR \rightarrow Y$ ；





- (3)执行周期
 - 执行周期完成加法运算，并将结果写回主存。
 - ① R_{1out} 和 ALU_{in} 有效，同时CU向ALU发“ADD”控制信号，使R1的内容和Y的内容相加，结果送寄存器Z中，记作 $(R_1)+Y \rightarrow Z$ ；
 - ② Z_{out} 和 MDR_{in} 有效，将运算结果送MDR，记作 $(Z) \rightarrow MDR$ ；
 - ③ 向主存发写命令，记作Write。





- 2.转移指令JCA
 - 这是一条条件转移指令，若上次运算结果有进位（ $C=1$ ），就转移；若上次运算结果无进位（ $C=0$ ），就顺序执行下一条指令。设A为位移量，转移地址等于PC的内容加位移量。
- (1)取指周期
 - 与上条指令的微操作序列完全相同。





- (2) 执行周期
 - 如果有进位 ($C=1$)，则完成 $(PC)+A \rightarrow PC$ 的操作，否则跳过以下几步。
 - ① PC_{out} 和 Y_{in} 有效，记作 $(PC) \rightarrow Y$ ($C=1$)；
 - ② Ad IR_{out} 和 ALU_{in} 有效，同时CU向ALU发“ADD”控制信号，使IR中的地址码字段A和Y的内容相加，结果送寄存器Z，记作 $Ad(IR)+Y \rightarrow Z$ ($C=1$)；
 - ③ Z_{out} 和 PC_{in} 有效，将运算结果送PC，记作 $(Z) \rightarrow PC$ ($C=1$)。



THANK YOU

